retargeting linux Search Adva Prefe

Web Results **1** - **10** of about **113** for **retargeting linux**. (**0.17** seconds)

Re: [**Linux**-aus] LA GST, and a TODO list for helpers
You could create proposals for restructuring or **retargeting Linux** Australia.
Stick to the concrete, positive actions, and your role in them, ...
lists.**linux**.org.au/archives/ **linux**-aus/2003-February/msg00138.html - 8k - Cached

LinuxCAD from Software Forge Inc. is the best CAD for **Linux**
LinuxCAD is available for **Linux**, SCO Open Server & Unixware, ... with new entity types and **retargeting** of the package for different application markets, ...
www.softwareforge.com/**linux**.html - 18k - Cached

MontaVista counters **Linux** tools shortage 'FUD'
Comprehensive information and resources on using **Linux** in embedded applications.
... "Now, Wind River and others are **retargeting** their legacy shelfware and ...
linuxdevices.com/news/NS9167108570.html - 49k - Cached

Wasabi Systems BSD vs. **Linux**
With **Linux**, however, device driver code must be reworked for every new architecture.
As a consequence, in recent porting efforts by NetBSD and **Linux** ...
www.wasabisystems.com/gpl/**linux**.htm - 42k - Cached

Linux Today - iTWeb: Surprising Forecast for **Linux**
... most common OS, by a lot. Tracking new spending misses **retargeting** last generation HW to **Linux**. **Linux** runs well on last generation ...
linuxtoday.com/ news_story.php3?ltsn=2004-04-05-037-26-OP-MR-SV-0013 - 36k - Supplemental Result - Cached

Slashdot | IBM Desktop **Linux** Pledge, One Year Later
IBM Desktop **Linux** Pledge, One Year Later -- article related to **Linux** ...
Retargeting large pieces of software is not something that happens over night. ...
linux.slashdot.org/article.pl?sid=05/ 01/26/0022201&tid=163&tid=136&tid=106 - 216k - Cached

Touchscreen driver, Xinput extension, input device interface, etc....
This is a generic problem to **Linux**; it is not a ARM problem at all (even if it
... input devices in the future on **Linux**, and will be **retargeting** the iPAQ's ...
lists.arm.**linux**.org.uk/pipermail/ **linux**-arm-kernel/2002-January/006978.html - 8k - Cached

Linux Kernel: Re: userspace irq balancer
... actually have a useful interaction with explicit irq balancing via **retargeting** IO-APIC RTE ... That doesn't really agree with the **Linux** model and is undesirable in ...
seclists.org/lists/**linux**-kernel/2003/May/5761.html - 12k - Supplemental Result - Cached

Linux Kernel: Re: Future **Linux** devel. Kernels
... Kernels"; Next in thread: Michael H. Warfield: "Re: Future **Linux** devel. ... I personally like the idea of simply **retargeting** the syslog to both local storage AND a ...
seclists.org/lists/**linux**-kernel/2000/May/2893.html - 13k - Supplemental Result - Cached
[More results from seclists.org]

MontaVista Software - Platform to Innovate
The **Linux** Trace Toolkit is a powerful Open Source tool for analyzing system ...
enabling LTT for cross development and **retargeting** LTT architecture support ...
www.mvista.com/products/tech/**linux**_trace_toolkit.html - 12k - Cached

Goooooooooogle ▶
Result Page: **1** 2 3 4 5 6 7 8 9 10 **Next**

retargeting linux Search

Search within results | Language Tools | Search Tips

Try your query on the entire web

Google Home - Advertising Programs - Business Solutions - About Google

patch-2.4.15 linux/arch/sparc64/kernel/irq.c

Next file: linux/arch/sparc64/kernel/isa.c
Previous file: linux/arch/sparc64/kernel/ioctl32.c
Back to the patch index
Back to the overall index

- Lines: 601
- Date: *Tue Nov 13 09:16:05 2001*
- Orig file: `v2.4.14/linux/arch/sparc64/kernel/irq.c`
- Orig date: *Tue Jul 3 17:08:19 2001*

```
diff -u --recursive --new-file v2.4.14/linux/arch/sparc64/kernel/irq.c linux/arch/sp
@@ -1,4 +1,4 @@
-/* $Id: irq.c,v 1.101 2001/06/04 06:50:18 ecd Exp $
+/* $Id: irq.c,v 1.109 2001/11/12 22:22:37 davem Exp $
  * irq.c: UltraSparc IRQ handling/init/registry.
  *
  * Copyright (C) 1997  David S. Miller  (davem@caip.rutgers.edu)
@@ -14,9 +14,10 @@
 #include <linux/mm.h>
 #include <linux/interrupt.h>
 #include <linux/slab.h>
-#include <linux/random.h> /* XXX ADD add_foo_randomness() calls... -DaveM */
+#include <linux/random.h>
 #include <linux/init.h>
 #include <linux/delay.h>
+#include <linux/proc_fs.h>

 #include <asm/ptrace.h>
 #include <asm/processor.h>
@@ -32,10 +33,7 @@
 #include <asm/hardirq.h>
 #include <asm/softirq.h>
 #include <asm/starfire.h>
-
-/* Internal flag, should not be visible elsewhere at all. */
-#define SA_IMAP_MASKED          0x100
-#define SA_DMA_SYNC             0x200
+#include <asm/uaccess.h>

 #ifdef CONFIG_SMP
 static void distribute_irqs(void);
@@ -64,7 +62,7 @@
 #endif

 #ifdef CONFIG_PCI
-/* This is a table of physical addresses used to deal with SA_DMA_SYNC.
+/* This is a table of physical addresses used to deal with IBF_DMA_SYNC.
  * It is used for PCI only to synchronize DMA transfers with IRQ delivery
  * for devices behind busses other than APB on Sabre systems.
  *
@@ -89,6 +87,23 @@
          NULL, NULL, NULL, NULL, NULL, NULL , NULL, NULL
 };
```

```
+static void register_irq_proc (unsigned int irq);
+
+/*
+ * Upper 2b of irqaction->flags holds the ino.
+ * irqaction->mask holds the smp affinity information.
+ */
+#define put_ino_in_irqaction(action, irq) \
+       action->flags &= 0xffffffffffffUL; \
+       if (__bucket(irq) == &pil0_dummy_bucket) \
+               action->flags |= 0xdeadUL << 48;  \
+       else \
+               action->flags |= __irq_ino(irq) << 48;
+#define get_ino_in_irqaction(action)   (action->flags >> 48)
+
+#define put_smpaff_in_irqaction(action, smpaff)        (action)->mask = (smpaff)
+#define get_smpaff_in_irqaction(action)        ((action)->mask)
+
 int get_irq_list(char *buf)
 {
        int i, len = 0;
@@ -108,13 +123,11 @@
                        len += sprintf(buf + len, "%10u ",
                                       kstat.irqs[cpu_logical_map(j)][i]);
 #endif
-               len += sprintf(buf + len, "%c %s",
-                              (action->flags & SA_INTERRUPT) ? '+' : ' ',
-                              action->name);
+               len += sprintf(buf + len, " %s:%lx", action->name, \
+                                         get_ino_in_irqaction(action));
                for(action = action->next; action; action = action->next) {
-                       len += sprintf(buf+len, ",%s %s",
-                                      (action->flags & SA_INTERRUPT) ? " +" : "",
-                                      action->name);
+                       len += sprintf(buf+len, ", %s:%lx", action->name, \
+                                         get_ino_in_irqaction(action));
                }
                len += sprintf(buf + len, "\n");
        }
@@ -286,23 +299,18 @@
        if(!handler)
            return -EINVAL;

-       if (!bucket->pil)
-               irqflags &= ~SA_IMAP_MASKED;
-       else {
-               irqflags |= SA_IMAP_MASKED;
-               if (bucket->flags & IBF_PCI) {
-                       /*
-                        * PCI IRQs should never use SA_INTERRUPT.
-                        */
-                       irqflags &= ~(SA_INTERRUPT);
-
-                       /*
-                        * Check wether we _should_ use DMA Write Sync
-                        * (for devices behind bridges behind APB).
-                        */
-                       if (bucket->flags & IBF_DMA_SYNC)
-                               irqflags |= SA_DMA_SYNC;
-               }
```

```
+       if ((bucket != &pil0_dummy_bucket) && (irqflags & SA_SAMPLE_RANDOM)) {
+               /*
+                * This function might sleep, we want to call it first,
+                * outside of the atomic block. In SA_STATIC_ALLOC case,
+                * random driver's kmalloc will fail, but it is safe.
+                * If already initialized, random driver will not reinit.
+                * Yes, this might clear the entropy pool if the wrong
+                * driver is attempted to be loaded, without actually
+                * installing a new handler, but is this really a problem,
+                * only the sysadmin is able to do this.
+                */
+               rand_initialize_irq(irq);
        }

        save_and_cli(flags);
@@ -316,12 +324,6 @@
                        restore_flags(flags);
                        return -EBUSY;
                }
-               if((action->flags & SA_INTERRUPT) ^ (irqflags & SA_INTERRUPT)) {
-                       printk("Attempt to mix fast and slow interrupts on IRQ%d "
-                              "denied\n", bucket->pil);
-                       restore_flags(flags);
-                       return -EBUSY;
-               }
                action = NULL;          /* Or else! */
        }

@@ -344,7 +346,7 @@
                return -ENOMEM;
        }

-       if ((irqflags & SA_IMAP_MASKED) == 0) {
+       if (bucket == &pil0_dummy_bucket) {
                bucket->irq_info = action;
                bucket->flags |= IBF_ACTIVE;
        } else {
@@ -405,12 +407,13 @@
                        bucket->pending = 0;
        }

-       action->mask = (unsigned long) bucket;
        action->handler = handler;
        action->flags = irqflags;
        action->name = name;
        action->next = NULL;
        action->dev_id = dev_id;
+       put_ino_in_irqaction(action, irq);
+       put_smpaff_in_irqaction(action, 0);

        if(tmp)
                tmp->next = action;
@@ -425,6 +428,8 @@
                set_softint(1 << bucket->pil);
        }
        restore_flags(flags);
+       if ((bucket != &pil0_dummy_bucket) && (!(irqflags & SA_STATIC_ALLOC)))
+               register_irq_proc(__irq_ino(irq));

 #ifdef CONFIG_SMP
```

```
        distribute_irqs();
@@ -492,7 +497,7 @@
        else
                *(bucket->pil + irq_action) = action->next;

-       if(action->flags & SA_IMAP_MASKED) {
+       if (bucket != &pil0_dummy_bucket) {
                unsigned long imap = bucket->imap;
                void **vector, *orig;
                int ent;
@@ -716,36 +721,80 @@
 /* Tune this... */
 #define FORWARD_VOLUME         12

-void handler_irq(int irq, struct pt_regs *regs)
-{
-       struct ino_bucket *bp, *nbp;
-       int cpu = smp_processor_id();
 #ifdef CONFIG_SMP
-       int should_forward = (this_is_starfire == 0     &&
-                             irq < 10                  &&
-                             current->pid != 0);
-       unsigned int buddy = 0;
+
+static inline void redirect_intr(int cpu, struct ino_bucket *bp)
+{
+       /* Ok, here is what is going on:
+        * 1) Retargeting IRQs on Starfire is very
+        *    expensive so just forget about it on them.
+        * 2) Moving around very high priority interrupts
+        *    is a losing game.
+        * 3) If the current cpu is idle, interrupts are
+        *    useful work, so keep them here.  But do not
+        *    pass to our neighbour if he is not very idle.
+        * 4) If sysadmin explicitly asks for directed intrs,
+        *    Just Do It.
+        */
+       struct irqaction *ap = bp->irq_info;
+       unsigned long cpu_mask = get_smpaff_in_irqaction(ap);
+       unsigned int buddy, ticks;
+
+       if (cpu_mask == 0)
+               cpu_mask = ~0UL;
+
+       if (this_is_starfire != 0 ||
+           bp->pil >= 10 || current->pid == 0)
+               goto out;

        /* 'cpu' is the MID (ie. UPAID), calculate the MID
         * of our buddy.
         */
-       if (should_forward != 0) {
-               buddy = cpu_number_map(cpu) + 1;
+       buddy = cpu_number_map(cpu) + 1;
+       if (buddy >= NR_CPUS ||
+           cpu_logical_map(buddy) == -1)
+               buddy = 0;
+
+       ticks = 0;
+       while ((cpu_mask & (1UL << buddy)) == 0) {
```

```
+               buddy++;
                if (buddy >= NR_CPUS ||
-                   (buddy = cpu_logical_map(buddy)) == -1)
+                   cpu_logical_map(buddy) == -1)
                        buddy = cpu_logical_map(0);
+               if (++ticks > NR_CPUS) {
+                       put_smpaff_in_irqaction(ap, 0);
+                       goto out;
+               }
+       }

-               /* Voo-doo programming. */
-               if (cpu_data[buddy].idle_volume < FORWARD_VOLUME)
-                       should_forward = 0;
+       if (buddy == cpu_number_map(cpu))
+               goto out;
+
+       buddy = cpu_logical_map(buddy);
+
+       /* Voo-doo programming. */
+       if (cpu_data[buddy].idle_volume < FORWARD_VOLUME)
+               goto out;
+
+       /* This just so happens to be correct on Cheetah
+        * at the moment.
+        */
+       buddy <<= 26;
+
+       /* Push it to our buddy. */
+       upa_writel(buddy | IMAP_VALID, bp->imap);
+
+out:
+       return;
+}

-               /* This just so happens to be correct on Cheetah
-                * at the moment.
-                */
-               buddy <<= 26;
-       }
 #endif

+void handler_irq(int irq, struct pt_regs *regs)
+{
+       struct ino_bucket *bp, *nbp;
+       int cpu = smp_processor_id();
+
 #ifndef CONFIG_SMP
        /*
         * Check for TICK_INT on level 14 softint.
@@ -765,6 +814,8 @@
                clear_softint(clr_mask);
        }
 #else
+       int should_forward = 1;
+
        clear_softint(1 << irq);
 #endif

@@ -781,6 +832,7 @@
```

```
 #endif
        for ( ; bp != NULL; bp = nbp) {
                unsigned char flags = bp->flags;
+               unsigned char random = 0;

                nbp = __bucket(bp->irq_chain);
                bp->irq_chain = 0;
@@ -795,34 +847,30 @@
                        if ((flags & IBF_MULTI) == 0) {
                                struct irqaction *ap = bp->irq_info;
                                ap->handler(__irq(bp), ap->dev_id, regs);
+                               random |= ap->flags & SA_SAMPLE_RANDOM;
                        } else {
                                void **vector = (void **)bp->irq_info;
                                int ent;
                                for (ent = 0; ent < 4; ent++) {
                                        struct irqaction *ap = vector[ent];
-                                       if (ap != NULL)
+                                       if (ap != NULL) {
+                                               ap->handler(__irq(bp), ap->dev_id, r
+                                               random |= ap->flags & SA_SAMPLE_RAND
+                                       }
                                }
                        }
                        /* Only the dummy bucket lacks IMAP/ICLR. */
                        if (bp->pil != 0) {
 #ifdef CONFIG_SMP
-                               /* Ok, here is what is going on:
-                                * 1) Retargeting IRQs on Starfire is very
-                                *    expensive so just forget about it on them.
-                                * 2) Moving around very high priority interrupts
-                                *    is a losing game.
-                                * 3) If the current cpu is idle, interrupts are
-                                *    useful work, so keep them here.  But do not
-                                *    pass to our neighbour if he is not very idle.
-                                */
-                               if (should_forward != 0) {
-                                       /* Push it to our buddy. */
+                               if (should_forward) {
+                                       redirect_intr(cpu, bp);
+                                       should_forward = 0;
                                        upa_writel(buddy | IMAP_VALID, bp->imap);
                                }
 #endif
                                upa_writel(ICLR_IDLE, bp->iclr);
+                               /* Test and add entropy */
+                               if (random)
+                                       add_interrupt_randomness(irq);
                        }
                } else
                        bp->pending = 1;
@@ -843,7 +891,7 @@
        kstat.irqs[cpu][irq]++;

        *(irq_work(cpu, irq)) = 0;
-       bucket = (struct ino_bucket *)action->mask;
+       bucket = get_ino_in_irqaction(action) + ivector_table;

        floppy_interrupt(irq, dev_cookie, regs);
        upa_writel(ICLR_IDLE, bucket->iclr);
```

```
@@ -896,10 +944,6 @@
                return -EINVAL;
        }

-       /* Only IMAP style interrupts can be registered as fast. */
-       if(bucket->pil == 0)
-               return -EINVAL;
-
        if(!handler)
                return -EINVAL;

@@ -918,6 +962,10 @@
                return -EBUSY;
        }

+       /*
+        * We do not check for SA_SAMPLE_RANDOM in this path. Neither do we
+        * support smp intr affinity in this path.
+        */
        save_and_cli(flags);
        if(irqflags & SA_STATIC_ALLOC) {
                if(static_irq_count < MAX_STATIC_ALLOC)
@@ -938,12 +986,13 @@
        bucket->irq_info = action;
        bucket->flags |= IBF_ACTIVE;

-       action->mask = (unsigned long) bucket;
        action->handler = handler;
-       action->flags = irqflags | SA_IMAP_MASKED;
+       action->flags = irqflags;
        action->dev_id = NULL;
        action->name = name;
        action->next = NULL;
+       put_ino_in_irqaction(action, irq);
+       put_smpaff_in_irqaction(action, 0);

        *(bucket->pil + irq_action) = action;
        enable_irq(irq);
@@ -993,7 +1042,7 @@
 #endif

        /* Register IRQ handler. */
-       err = request_irq(build_irq(0, 0, 0UL, 0UL), cfunc, (SA_INTERRUPT | SA_STATI
+       err = request_irq(build_irq(0, 0, 0UL, 0UL), cfunc, SA_STATIC_ALLOC,
                          "timer", NULL);

        if(err) {
@@ -1079,14 +1128,10 @@
 #ifdef CONFIG_SMP
 static int retarget_one_irq(struct irqaction *p, int goal_cpu)
 {
-       struct ino_bucket *bucket = __bucket(p->mask);
+       struct ino_bucket *bucket = get_ino_in_irqaction(p) + ivector_table;
        unsigned long imap = bucket->imap;
        unsigned int tid;

-       /* Never change this, it causes problems on Ex000 systems. */
-       if (bucket->pil == 12)
-               return goal_cpu;
-
```

```
        if (tlb_type == cheetah) {
                tid = __cpu_logical_map[goal_cpu] << 26;
                tid &= IMAP_AID_SAFARI;
@@ -1114,11 +1159,16 @@

        save_and_cli(flags);
        cpu = 0;
-       for(level = 0; level < NR_IRQS; level++) {
+
+       /*
+        * Skip the timer at [0], and very rare error/power intrs at [15].
+        * Also level [12], it causes problems on Ex000 systems.
+        */
+       for(level = 1; level < NR_IRQS; level++) {
                struct irqaction *p = irq_action[level];
+               if (level == 12) continue;
                while(p) {
-                       if(p->flags & SA_IMAP_MASKED)
-                               cpu = retarget_one_irq(p, cpu);
+                       cpu = retarget_one_irq(p, cpu);
                        p = p->next;
                }
        }
@@ -1228,7 +1278,173 @@
                             : "g1");
 }

-void init_irq_proc(void)
+static struct proc_dir_entry * root_irq_dir;
+static struct proc_dir_entry * irq_dir [NUM_IVECS];
+
+#ifdef CONFIG_SMP
+
+#define HEX_DIGITS 16
+
+static unsigned int parse_hex_value (const char *buffer,
+               unsigned long count, unsigned long *ret)
 {
-       /* For now, nothing... */
+       unsigned char hexnum [HEX_DIGITS];
+       unsigned long value;
+       int i;
+
+       if (!count)
+               return -EINVAL;
+       if (count > HEX_DIGITS)
+               count = HEX_DIGITS;
+       if (copy_from_user(hexnum, buffer, count))
+               return -EFAULT;
+
+       /*
+        * Parse the first 8 characters as a hex string, any non-hex char
+        * is end-of-string. '00e1', 'e1', '00E1', 'E1' are all the same.
+        */
+       value = 0;
+
+       for (i = 0; i < count; i++) {
+               unsigned int c = hexnum[i];
+
+               switch (c) {
```

```
+                       case '0' ... '9': c -= '0'; break;
+                       case 'a' ... 'f': c -= 'a'-10; break;
+                       case 'A' ... 'F': c -= 'A'-10; break;
+               default:
+                       goto out;
+               }
+               value = (value << 4) | c;
+       }
+out:
+       *ret = value;
+       return 0;
 }
+
+static unsigned long hw_to_logical(unsigned long mask)
+{
+       unsigned long new_mask = 0UL;
+       int i;
+
+       for (i = 0; i < NR_CPUS; i++) {
+               if (mask & (1UL << i)) {
+                       int logical = cpu_number_map(i);
+
+                       new_mask |= (1UL << logical);
+               }
+       }
+
+       return new_mask;
+}
+
+static unsigned long logical_to_hw(unsigned long mask)
+{
+       unsigned long new_mask = 0UL;
+       int i;
+
+       for (i = 0; i < NR_CPUS; i++) {
+               if (mask & (1UL << i)) {
+                       int hw = cpu_logical_map(i);
+
+                       new_mask |= (1UL << hw);
+               }
+       }
+
+       return new_mask;
+}
+
+static int irq_affinity_read_proc (char *page, char **start, off_t off,
+                       int count, int *eof, void *data)
+{
+       struct ino_bucket *bp = ivector_table + (long)data;
+       struct irqaction *ap = bp->irq_info;
+       unsigned long mask = get_smpaff_in_irqaction(ap);
+
+       mask = logical_to_hw(mask);
+
+       if (count < HEX_DIGITS+1)
+               return -EINVAL;
+       return sprintf (page, "%016lx\n", mask == 0 ? ~0UL : mask);
+}
+
+static inline void set_intr_affinity(int irq, unsigned long hw_aff)
```

```
+{
+       struct ino_bucket *bp = ivector_table + irq;
+       unsigned long aff = hw_to_logical(hw_aff);
+
+       /*
+        * Users specify affinity in terms of cpu ids, which is what
+        * is displayed via /proc/cpuinfo. As soon as we do this,
+        * handler_irq() might see and take action.
+        */
+       put_smpaff_in_irqaction((struct irqaction *)bp->irq_info, aff);
+
+       /* Migration is simply done by the next cpu to service this
+        * interrupt.
+        */
+}
+
+static int irq_affinity_write_proc (struct file *file, const char *buffer,
+                                    unsigned long count, void *data)
+{
+       int irq = (long) data, full_count = count, err;
+       unsigned long new_value;
+
+       err = parse_hex_value(buffer, count, &new_value);
+
+       /*
+        * Do not allow disabling IRQs completely - it's a too easy
+        * way to make the system unusable accidentally :-) At least
+        * one online CPU still has to be targeted.
+        */
+       new_value &= cpu_online_map;
+       if (!new_value)
+               return -EINVAL;
+
+       set_intr_affinity(irq, new_value);
+
+       return full_count;
+}
+
+#endif
+
+#define MAX_NAMELEN 10
+
+static void register_irq_proc (unsigned int irq)
+{
+       char name [MAX_NAMELEN];
+
+       if (!root_irq_dir || irq_dir[irq])
+               return;
+
+       memset(name, 0, MAX_NAMELEN);
+       sprintf(name, "%x", irq);
+
+       /* create /proc/irq/1234 */
+       irq_dir[irq] = proc_mkdir(name, root_irq_dir);
+
+#ifdef CONFIG_SMP
+       /* XXX SMP affinity not supported on starfire yet. */
+       if (this_is_starfire == 0) {
+               struct proc_dir_entry *entry;
+
```

```
+               /* create /proc/irq/1234/smp_affinity */
+               entry = create_proc_entry("smp_affinity", 0600, irq_dir[irq]);
+
+               if (entry) {
+                       entry->nlink = 1;
+                       entry->data = (void *)(long)irq;
+                       entry->read_proc = irq_affinity_read_proc;
+                       entry->write_proc = irq_affinity_write_proc;
+               }
+       }
+#endif
+}
+
+void init_irq_proc (void)
+{
+       /* create /proc/irq */
+       root_irq_dir = proc_mkdir("irq", 0);
+}
+
```

FUNET's LINUX-ADM group, linux-adm@nic.funet.fi
TCL-scripts by Sam Shen (who was at: slshen@lbl.gov)

```
Set     Items   Description
S1         93   (RELOCABLE OR RELOCAT?)(N)(LINKING OR LINKER?) OR COFF(2N)-
               RELOC?
S2         56   RD (unique items)
S3         41   S2 NOT PY>1999
S4         37   S3 NOT PD=19990101:20010101
S5         37   S4 NOT PD=20010101:20030508
File   2:INSPEC 1969-2003/Apr W4
         (c) 2003 Institution of Electrical Engineers
File   5:Biosis Previews(R) 1969-2003/May W1
         (c) 2003 BIOSIS
File   7:Social SciSearch(R) 1972-2003/Apr W4
         (c) 2003 Inst for Sci Info
File   8:Ei Compendex(R) 1970-2003/Apr W4
         (c) 2003 Elsevier Eng.  Info. Inc.
File  11:PsycINFO(R) 1887-2003/May W1
         (c) 2003 Amer. Psychological Assn.
File  15:ABI/Inform(R) 1971-2003/May 07
         (c) 2003 ProQuest Info&Learning
File  16:Gale Group PROMT(R) 1990-2003/May 06
         (c) 2003 The Gale Group
File  20:Dialog Global Reporter 1997-2003/May 07
         (c) 2003 The Dialog Corp.
File  73:EMBASE 1974-2003/Apr W4
         (c) 2003 Elsevier Science B.V.
File  88:Gale Group Business A.R.T.S. 1976-2003/May 06
         (c) 2003 The Gale Group
File 144:Pascal 1973-2003/Apr W4
         (c) 2003 INIST/CNRS
File 148:Gale Group Trade & Industry DB 1976-2003/May 06
         (c)2003 The Gale Group
File 155:MEDLINE(R) 1966-2003/May W1
         (c) format only 2003 The Dialog Corp.
File 156:ToxFile 1965-2003/May W1
         (c) format only 2003 The Dialog Corporation
File 160:Gale Group PROMT(R) 1972-1989
         (c) 1999 The Gale Group
File 194:FBODaily 1982/Dec-2003/Jan
         (c) format only 2003 The Dialog Corp.
File 275:Gale Group Computer DB(TM) 1983-2003/May 06
         (c) 2003 The Gale Group
File 347:JAPIO Oct 1976-2002/Dec(Updated 030402)
         (c) 2003 JPO & JAPIO
File 348:EUROPEAN PATENTS 1978-2003/Apr W04
         (c) 2003 European Patent Office
File 349:PCT FULLTEXT 1979-2002/UB=20030501,UT=20030424
         (c) 2003 WIPO/Univentio
File 351:Derwent WPI 1963-2003/UD,UM &UP=200329
         (c) 2003  Thomson Derwent
File 440:Current Contents Search(R) 1990-2003/May 07
         (c) 2003 Inst for Sci Info
File 519:D&B-Duns Finan.Records Plus(TM) 2003/Jan
         (c) 2003 Dun & Bradstreet
File 610:Business Wire 1999-2003/May 07
         (c) 2003 Business Wire.
File 613:PR Newswire 1999-2003/May 07
         (c) 2003 PR Newswire Association Inc
File 621:Gale Group New Prod.Annou.(R) 1985-2003/May 06
         (c) 2003 The Gale Group
File 636:Gale Group Newsletter DB(TM) 1987-2003/May 06
         (c) 2003 The Gale Group
File 649:Gale Group Newswire ASAP(TM) 2003/May 06
         (c) 2003 The Gale Group
File 654:US PAT.FULL. 1976-2003/May 06
         (c) FORMAT ONLY 2003 THE DIALOG CORP.
File 810:Business Wire 1986-1999/Feb 28
         (c) 1999 Business Wire
File 995:NewsRoom 2000
```

5/6/1 (Item 1 from file: 2)
6145199 INSPEC Abstract Number: C1999-03-6115-001
Title: The architecture of Montana: an open and extensible programming environment with an incremental C++ compiler
Publication Date: Nov. 1998

5/6/2 (Item 2 from file: 2)
4913521 INSPEC Abstract Number: C9505-6150C-014
Title: GPXA: A general purpose cross-assembler
Publication Date: Sept.-Dec. 1994

5/6/3 (Item 3 from file: 2)
02686774 INSPEC Abstract Number: C86033863
Title: STEP Development Tools: METASTEP language system
Publication Date: 1985

5/6/4 (Item 4 from file: 2)
02237375 INSPEC Abstract Number: C84022570
Title: M29-an advanced retargetable microcode assembler
Publication Date: Oct. 1983

5/6/5 (Item 5 from file: 2)
01846892 INSPEC Abstract Number: C82018883
Title: Toward user sharing of the microprogramming level under UNIX on the Perkin-Elmer 3220
Publication Date: 1981

5/6/6 (Item 6 from file: 2)
01629367 INSPEC Abstract Number: C81005401
Title: The 64000 linker
Publication Date: Oct. 1980

5/6/7 (Item 7 from file: 2)
00960606 INSPEC Abstract Number: C76024238
Title: A portable linking loader
Publication Date: 1976

5/6/8 (Item 8 from file: 2)
00663470 INSPEC Abstract Number: C74017654
Title: STARAN/RADCAP system software
Publication Date: 1973

5/6/9 (Item 1 from file: 5)
11657662 BIOSIS NO.: 199800439393
Frequent childhood geographic relocation: Its impact on drug use initiation and the development of alcohol and other drug-related problems among adolescents and young adults.
1998

5/6/10 (Item 1 from file: 8)
01137403
Title: DESIGNING AND BUILDING MICROCOMPUTER-BASED SYSTEMS.
Publication Year: 1981

5/6/11 (Item 2 from file: 8)
01094595

Title: DESIGN OF A UNIVERSAL MICROPROCESSOR LINKER.
Publication Year: 1980

5/6/12 (Item 1 from file: 16)
05981882 Supplier Number: 53332233 (USE FORMAT 7 FOR FULLTEXT)
Crossware's New ANSI C Compiler Supports Motorola's Latest ColdFire Chips.
Dec 2, 1998
Word Count: 367

5/6/13 (Item 2 from file: 16)
05259718 Supplier Number: 48014678 (USE FORMAT 7 FOR FULLTEXT)
New C Compiler for PIC17CXXX MCUs Provides Powerful Integration and Ease of Use.
Sept 30, 1997
Word Count: 507

5/6/14 (Item 3 from file: 16)
04750760 Supplier Number: 46992621 (USE FORMAT 7 FOR FULLTEXT)
Chips: IAR Systems Releases Embedded Toolset for Mitsubishi's New Advanced 16-Bit M16C Microcontroller Family
Dec 23, 1996
Word Count: 897

5/6/15 (Item 1 from file: 88)
01499855 SUPPLIER NUMBER: 00521618
The Microprocessor Universal Format for Object Modules Proposed Standard.
July-Aug., 1983

5/6/16 (Item 1 from file: 160)
02144898
NEW MICRO-10 SOFTWARE FROM MICROPLOT
January 11, 1989

5/6/17 (Item 2 from file: 160)
01852645
American Automation Ships First Cross-Development Tools for the Apple Macintosh
December 7, 1987

5/6/18 (Item 3 from file: 160)
01322428
Development system bows.
March 17, 1986

5/6/19 (Item 1 from file: 194)
2324800
EMULATION SYSTEM FOR MICROPROCESSOR
PUBLICATION DATE: JULY 31, 1989

5/6/20 (Item 2 from file: 194)
2184724
SUN MICROSYSTEMS INC POC
PUBLICATION DATE: JANUARY 13, 1989

5/6/21 (Item 1 from file: 275)
01863969 SUPPLIER NUMBER: 17496572 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Providing CAD object management services through a base class library. (HP PE/SolidDesigner CAD software) (includes related article on exception

handling) (Product Information) (Technical)
Oct, 1995
WORD COUNT: 8293 LINE COUNT: 00653

5/6/22 (Item 2 from file: 275)
01668763 SUPPLIER NUMBER: 15047570 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Peering inside the PE: a tour of the Win32 portable executable file format. (includes related articles on header formats, directory entries, header fields) (Tutorial)
March, 1994
WORD COUNT: 9524 LINE COUNT: 00734

5/6/23 (Item 3 from file: 275)
01432485 SUPPLIER NUMBER: 10767454 (USE FORMAT 7 OR 9 FOR FULL TEXT)
Text editors. (Programming On Purpose) (column)
May, 1991
WORD COUNT: 3289 LINE COUNT: 00239

5/6/24 (Item 4 from file: 275)
01095295 SUPPLIER NUMBER: 00541666
Review of RMA and RLINK.
March, 1984

5/6/25 (Item 1 from file: 347)
02851585
PICTURE PROCESSING METHOD

5/6/26 (Item 1 from file: 519)
0468727
PROPANE USA INC

DUNS NUMBER: 02-978-4840

HISTORY text available (04/15/02)

OPERATION text available (04/15/02)

Enter REPORT Sn/BIR, REPORT Sn/SER or REPORT Sn/PAR to receive special reports directly from D&B (see HELP BIR, HELP RATES 516)

5/6/27 (Item 1 from file: 621)
01041059 Supplier Number: 40052665 (USE FORMAT 7 FOR FULLTEXT)
SKY COMPUTERS INTRODUCES FIRST SOFTWARE FOR TEXAS INSTRUMENTS' TMS 320-20 DIGITAL SIGNAL PROCESSING (DSP) CHIP
May 11, 1987
Word Count: 666

5/6/28 (Item 2 from file: 621)
01022817 Supplier Number: 39709514 (USE FORMAT 7 FOR FULLTEXT)
TEKTRONIX INTRODUCES 32-BIT MICROPROCESSOR SUPPORT SYSTEMS
March 3, 1986
Word Count: 592

5/6/29 (Item 3 from file: 621)
01012316 Supplier Number: 39611078 (USE FORMAT 7 FOR FULLTEXT)
INTRODUCING THE WORLD'S FIRST 64-BIT-WIDE REAL-TIME ICE
Oct 15, 1985
Word Count: 869

5/6/30 (Item 4 from file: 621)
01007288 Supplier Number: 39568265 (USE FORMAT 7 FOR FULLTEXT)
MACRO-META ASSEMBLER ASSEMBLES TWICE AS FAST
August 5, 1985
Word Count: 332

5/6/31 (Item 1 from file: 636)
02992192 Supplier Number: 46103621 (USE FORMAT 7 FOR FULLTEXT)
MITSUBISHI: Mitsubishi announces comprehensive 8bit IAR Microcontroller development kit
Jan 30, 1996
Word Count: 678

5/6/32 (Item 1 from file: 654)
3983545 **IMAGE Available
Derwent Accession: 1998-161529
Utility
E/ **Download of interpreter to a printer**

Fulltext Word Count: 4336
Number of Claims: 7
Exemplary Claim Number: 1
Number of Drawing Sheets: 2
Number of Figures: 2
Number of US cited patent references: 4

5/6/33 (Item 2 from file: 654)
3849202 **IMAGE Available
Derwent Accession: 1993-388281
Utility
E/ **Dynamically configurable kernel; IN A COMPUTER SYSTEM**

Fulltext Word Count: 8629
Number of Claims: 18
Exemplary Claim Number: 13
Number of Drawing Sheets: 16
Number of Figures: 20
Number of US cited patent references: 8
Number of non-patent cited references: 5

5/6/34 (Item 3 from file: 654)
3849167 **IMAGE Available
Derwent Accession: 1997-297649
Utility
E/ **Software mechanism for accurately handling exceptions generated by speculatively scheduled instructions**

Fulltext Word Count: 7400
Number of Claims: 21
Exemplary Claim Number: 1
Number of Drawing Sheets: 10
Number of Figures: 11
Number of US cited patent references: 1

5/6/35 (Item 4 from file: 654)
3842709 **IMAGE Available
Derwent Accession: 1997-271643
Utility
E/ **Software mechanism for accurately handling exceptions generated by instructions scheduled speculatively due to branch elimination**

Fulltext Word Count: 7546
Number of Claims: 12

Exemplary Claim Number: 1
Number of Drawing Sheets: 10
Number of Figures: 11
Number of US cited patent references: 5

5/6/36 (Item 5 from file: 654)
3555581 **IMAGE Available
Derwent Accession: 1995-005623
Utility
M/ **Auxillary coin dispenser with transaction data recording and transfer mechanisms**

Fulltext Word Count: 4668
Number of Claims: 7
Exemplary Claim Number: 1
Number of Drawing Sheets: 2
Number of Figures: 2
Number of US cited patent references: 2

5/6/37 (Item 6 from file: 654)
3423483 **IMAGE Available
Derwent Accession: 1993-311958
Utility
E/ **Load time linker for software used with a multiprocessor system**

Fulltext Word Count: 6079
Number of Claims: 11
Exemplary Claim Number: 1
Number of Drawing Sheets: 4
Number of Figures: 4
Number of US cited patent references: 4

5/5/3 (Item 3 from file: 2)
DIALOG(R)File 2:INSPEC

02686774 INSPEC Abstract Number: C86033863
Title: STEP Development Tools: METASTEP language system
Author(s): Wilburn, D.L.; Schleimer, S.
Author Affiliation: STEP Eng. Inc., Sunnyvale, CA, USA
Conference Title: Proceedings of the 18th Annual Workshop on Microprogramming (Cat. No.85CH2232-7) p.157-64
Publisher: IEEE Comput. Soc. Press, Washington, DC, USA
Publication Date: 1985 Country of Publication: USA xiv+201 pp.
ISBN: 0 89791 172 5
U.S. Copyright Clearance Center Code: 0-89791-172-5/85/0012-0157$00.75
Conference Sponsor: IEEE; ACM; Euromicro
Conference Date: 3-6 Dec. 1985 Conference Location: Pacific Grove, CA, USA
Language: English Document Type: Conference Paper (PA)
Treatment: Practical (P)
Abstract: STEP Development Tools (SDT) is a general-purpose microprogram development system. The METASTEP language system is composed of four tools of the SDT needed to write microprograms: a definition processor, a retargetable assembler, a retargetable cross-assembler, and a **relocatable linker** . These tools are of commercial quality, providing complete languages, quality diagnostics, full interface to other support tools, and high performance. The language system supports microcode debug by providing complete run-time information and by interacting directly with other debug tools provided in SDT. (4 Refs)
Subfile: C
Descriptors: computer debugging; microprogramming; program assemblers
Identifiers: STEP Development Tools; METASTEP language system; general-purpose microprogram development system; definition processor; retargetable assembler; retargetable cross-assembler; **relocatable linker** ; quality diagnostics; full interface; support tools; microcode debug; run-time information; debug tools
Class Codes: C5220 (Computer architecture); C6150C (Compilers, interpreters and other processors)

5/5/4 (Item 4 from file: 2)
DIALOG(R)File 2:INSPEC

02237375 INSPEC Abstract Number: C84022570
Title: M29-an advanced retargetable microcode assembler
Author(s): Eager, M.J.
Author Affiliation: Advanced Micro Devices Inc., Sunnyvale, CA, USA
Conference Title: MICRO 16. Proceedings of the 16th Annual Microprogramming Workshop p.92-100
Publisher: IEEE Comput. Soc. Press, Silver Spring, MD, USA
Publication Date: Oct. 1983 Country of Publication: USA xvi+229 pp.
ISBN: 0 89791 114 8
U.S. Copyright Clearance Center Code: 0 89791 114 8/83/0010/0092$00.75
Conference Sponsor: IEEE; ACM
Conference Date: 11-14 Oct. 1983 Conference Location: Downingtown, PA, USA
Language: English Document Type: Conference Paper (PA)
Treatment: Practical (P)
Abstract: The M29 Assembler, a retargetable microcode assembler that produces relocatable microcode, is described. It provides a straightforward means of describing the microinstruction format, a very general method of specifying field contents, a comprehensive macro facility, and a multifunction **relocating linker** that permits overlays and libraries of microcode. Several features support improved productivity and reduction of coding errors. (13 Refs)
Subfile: C
Descriptors: microprogramming; program assemblers
Identifiers: M29 Assembler; retargetable microcode assembler; relocatable

microcode; microinstruction format
Class Codes: C6150C (Compilers, interpreters and other processors)

5/5/6 (Item 6 from file: 2)
DIALOG(R)File 2:INSPEC

01629367 INSPEC Abstract Number: C81005401
Title: The 64000 linker
Author(s): Stewart, J.B.
Author Affiliation: Hewlett-Packard Co., Colorado Springs, CO, USA
Journal: Hewlett-Packard Journal vol.31, no.10 p.25-6
Publication Date: Oct. 1980 Country of Publication: USA
CODEN: HPJOAX ISSN: 0018-1153
Language: English Document Type: Journal Paper (JP)
Treatment: Practical (P)
Abstract: Describes the linker used in the 64000 development system. The 64000 linker takes relocatable object files generated by the assembler or Pascal compiler and combines them to produce an executable absolute file. The linker resolves symbolic references between relocatable files (linking). It also assigns relocatable code to an absolute location in the target processor's logical address space and changes memory references to refer to the absolute memory locations (relocation). The linker was designed with three major goals: to support a wide variety of microprocessors, to be easy to use, and to provide the user with a complete set of features to facilitate **linking relocatable** modules for complex microprocessor systems. (0 Refs)
Subfile: C
Descriptors: program processors
Identifiers: 64000 linker; 64000 development system; linker; relocatable files; microprocessor systems; Hewlett Packard
Class Codes: C6150C (Compilers, interpreters and other processors)

5/5/8 (Item 8 from file: 2)
DIALOG(R)File 2:INSPEC

00663470 INSPEC Abstract Number: C74017654
Title: STARAN/RADCAP system software
Author(s): Davis, E.W.
Author Affiliation: Goodyear Aerospace Corp., Akron, OH, USA
Conference Title: Proceedings of the 1973 Sagamore Computer Conference on Parallel Processing p.153-9
Publisher: IEEE, New York, NY, USA
Publication Date: 1973 Country of Publication: USA vii+190 pp.
Conference Sponsor: IEEE; ACM; Syracuse Univ
Conference Date: 22-24 Aug. 1973 Conference Location: Sagamore, NY, USA
Language: English Document Type: Conference Paper (PA)
Treatment: Practical (P)
Abstract: System software is described for RADCAP, the operational associative array processor facility installed at Rome Air Development Center, N.Y. The description covers the software for the stand-alone operation of the Goodyear Aerospace STARAN associative array (parallel) processor, which is supported by a disk operating system with a macro-assembler, a **relocating linker** and loader, an interactive debug package, and control procedures. Also described is the software for the STARAN processor when integrated with the Honeywell Information Systems 645 sequential computer, which runs under the Multics time-shared operating system. (8 Refs)
Subfile: C
Descriptors: parallel processing
Identifiers: STARAN; RADCAP; system software; associative array
Class Codes: C6150J (Operating systems)

5/5/13 (Item 2 from file: 16)
DIALOG(R)File 16:Gale Group PROMT(R)

05259718 Supplier Number: 48014678 (USE FORMAT 7 FOR FULLTEXT)
New C Compiler for PIC17CXXX MCUs Provides Powerful Integration and Ease of Use.
Business Wire, p09300168
Sept 30, 1997
Language: English Record Type: Fulltext
Document Type: Newswire; Trade
Word Count: 507
PUBLISHER NAME: Business Wire
COMPANY NAMES: *Microchip Technology Inc.
EVENT NAMES: *336 (Product introduction)
GEOGRAPHIC NAMES: *1USA (United States)
PRODUCT NAMES: *3674100 (Integrated & Hybrid Circuits)
INDUSTRY NAMES: BUS (Business, General); BUSN (Any type of business)
NAICS CODES: 334413 (Semiconductor and Related Device Manufacturing)
TICKER SYMBOLS: MCHP
SPECIAL FEATURES: COMPANY

5/5/24 (Item 4 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)

01095295 SUPPLIER NUMBER: 00541666
Review of RMA and RLINK.
Dibble, P.
68 Micro Journal, v6, n3, p27-29
March, 1984
DOCUMENT TYPE: evaluaton ISSN: 0194-5025 LANGUAGE: ENGLISH
RECORD TYPE: ABSTRACT

ABSTRACT: RMA (Relocating Macro Assembler) is a program development tool that makes writing large programs easier. RLINK (**Relocating Linker**) takes the files created by RMA and converts them into executable form. These programs are bundled with the 'C' language from Microware. Both programs are important software to own if the user is serious about assembly language.

DESCRIPTORS: Application Development Software; Program Generators; Macro; Bundled Software; Assembly Language; Programming Language; C Programming Language
TRADE NAMES: Relocating Macro Assembler--evaluation; **Relocating Linker** --evaluation
FILE SEGMENT: CD File 275

5/5/32 (Item 1 from file: 654)
DIALOG(R)File 654:US PAT.FULL.

3983545 **IMAGE Available
Derwent Accession: 1998-161529
Utility
E/ **Download of interpreter to a printer**
Inventor: Rivers, Martin Geoffrey, Lexington, KY
Songer, Christopher Mark, Foster City, CA
Spears, Hugh Deral, Lexington, KY
Assignee: Lexmark International, Inc. (02), Lexington, KY
Lexmark International Inc (Code: 26474)
Examiner: Evans, Arthur G. (Art Unit: 272)
Combined Principal Attorneys: Brady, John A.

Publication Number	Kind	Date	Application Number	Filing Date

Main Patent	US 5754748	A	19980519	US 96713300	19960913
Priority				US 96713300	19960913

Current US Classification (Main): 358001170 (X-ref): 358001130
US Classification on document (Main): 395116 (X-ref): 395112
International Classification (Edition 1): G06K-015/00
Examiner Field of Search (US): 395101; 395112; 395114; 395115; 395116; 395117; 395892; 395882; 395884; 395893; 395894; 395828; 707514; 707509; 707515; 707516; 345515; 345516; 345517

Cited US Patents:

Patent Number	Date YYYYMM	Main US Class	Inventor
US 5179610	199301	395500	Ishikawa
US 5222200	199306	395112	Calister
US 5353388	199410	395117	Motoyama
US 5446837	199508	707514	Motoyama

Fulltext Word Count: 4336
Number of Claims: 7
Exemplary Claim Number: 1
Number of Drawing Sheets: 2
Number of Figures: 2
Number of US cited patent references: 4
Calculated Expiration Date: 20160913

Abstract:
Printer (1) has code in memory (5) to interpret one or more page description languages and also stores in memory (5) a table of routines useful in other interpreters and their addresses, as well as code to link partially compiled object code in response to directions in Relocation Tables in the partially compiled code. Data is downloaded to the printer by cable (23) to add another interpreter to the printer. That data is employed by the linking code, which responds to the directions in the Relocation Tables to complete the new interpreter making use of routines already stored in the printer. Since the stored routine table is lengthy, it is compressed. The routine table may be downloaded rather than permanently stored.

What is claimed is:

1. A printer having a microprocessor which controls said printer and a memory which permanently stores data processing code to interpret at least a first page description language characterized by said memory having additional stored code comprising a table of routines and their addresses which may be used in a second page description language interpreter, and comprising data processing code to link downloaded raw data and downloaded related linking information using said table of routines as directed by said related linking information to form said second page description language interpreter. (Main Claim)
2. The printer as in claim 1 in which said table of routines is compressed by storing for the names of each of said routines the result of arithmetic operations on each name of said routines in said table.
3. The printer as in claim 2 in which said table of routines is further compressed by storing as the address of each said routines a designation of one of more than 32 base numbers substrated from each address of said routines and the difference between the base number designated and each said address.
4. The printer as in claim 3 in which the numerical value assigned to each character of the names of said routines are separated by at least the value of two.
5. The printer as in claim 2 in which the numerical value assigned to each character of the names of said routines are separated by a least the value of two.
6. The method of adding function to an existing printer operative to

interpret at least a first page description language to interpret a second page description language comprising preparing data processing code to interpret said second page description language which requires linking with existing routines in said printer, entering said prepared code into said printer, linking said prepared code in said printer using a table of routines and their location stored in memory in said printer to form an interpreter for said second page description language and storing said linked code in permanent memory of said printer.

7. The method as in claim 6 in which said table of routines is downloaded into said memory in said printer.

5/5/33 (Item 2 from file: 654)

3849202 **IMAGE Available
Derwent Accession: 1993-388281
Utility
E/ **Dynamically configurable kernel; IN A COMPUTER SYSTEM**

Inventor: Allen, Tom, Marlborough, MA
Provino, Joseph E., Cambridge, MA
Pittore, William F., Lexington, MA
Assignee: Sun Microsystems, Inc. (02), Mountain View, CA
Sun Microsystems Inc (Code: 24836)
Examiner: Kriess, Kevin A. (Art Unit: 236)
Assistant Examiner: Toplu, Lucien
Law Firm: Blakely Sokoloff Taylor & Zafman LLP

	Publication Number	Kind	Date	Application Number	Filing Date
Main Patent	US 5634058	A	19970527	US 95540875	19951011
Continuation	Abandoned			US 92893337	19920603
Priority				US 95540875	19951011
				US 92893337	19920603

Current US Classification (Main): 717011000 (X-ref): 709331000
US Classification on document (Main): 395712 (X-ref): 3642808; 3642802; 3649754; 3649752; 395685
International Classification (Edition 1): G06F-009/445
Examiner Field of Search (US): 395700

Cited US Patents:

Patent Number	Date YYYYMM	Main US Class	Inventor
US 4445190	198404	364900	Pierschalla
US 4787034	198811	364300	Szoke
US 5175828	199212	395375	Hall
US 5226160	199307	395650	Waldron
US 5247678	199309	395700	Littleton
US 5291601	199403	395700	Sands
US 5303376	199404	395700	Taki
US 5303392	199404	395775	Carney

Cited non-Patent References:

Software Practice & Experience, vol. 21, No. 4, Apr. 1991, Chichester, GB; pp. 375-390 XP147180 W. Wilson Ho et al.: "An Approach to Genuine Dynamic Linking" *abstract* *p. 379, line 35--p. 384, line 13; figures 2,3.

Dr. Dobb's Journal of Software Tools, vol. 15, No. 5, May 1990, US; pp. 30-109 Gary Syck: "Dynamic Link Libraries for DOS" *p. 30, middle col. line 7--p. 32, left col., line 13* *p. 36, left col., line 10--p. 39, left col., line 1*.

Proceedings of the Spring 1990 EUUG Conference, 23 Apr. 1990, Munich, DE, pp. 133-138; Dieter Konnerth et al.: "Dynamic Driver Loading for Unix System V" *p. 133, line 27-line 32* *p. 135, line 8-line 40* *p. 136,

line 1-line 39.
Peacock, Dynamic Shared Libraries, UNIX Review, May 1991, V 9, n 5 P37(6).
Wirth et al, The Oberon System, Software Practice and Experience, vol. 19(9), Sep. 1989 pp. 857-893.

Fulltext Word Count: 8629
Number of Claims: 18
Exemplary Claim Number: 13
Number of Drawing Sheets: 16
Number of Figures: 20
Number of US cited patent references: 8
Number of non-patent cited references: 5
Calculated Expiration Date: 20120603

Abstract:
A dynamically configurable operating system is achieved by providing a module sub-system which intercepts requests by processes to access a module in the operating system and determines whether the module has been loaded in the kernel memory and linked the other modules located in the kernel memory and installed in the appropriate table. If the module has been into the kernel memory, and installed the module sub-system grants the requesting installed process access to the module and processing continues. If the module has not been loaded into the kernel memory, the module sub-system will retrieve a copy of the module stored and copy it into kernel memory. The module is then linked to the other modules located in the kernel and installed. Once the module is loaded and linked and installed, access is granted to the requesting process and normal processing continues.

We claim:

13. In a computer system comprising resources such as a Central Processing Unit (CPU), memory and input/output means, an operating system which executes processes and utilizes the resources, the operating system comprising at least one module loaded into operating system memory and installed as part of the operating system, a process for dynamically configuring the operating system by loading the modules into operating system memory and installing the loaded modules on an as needed basis comprising the steps of:
when a requesting process issues a request to the operating system to access a requested module;
a module subsystem intercepting the request to access the requested module such that the request is temporarily not executed by the operating system, and
determining if the requested module has been loaded and installed into the operating system by determining if a module configuration table stored in operating system memory contains an entry for the requested module having a value indicative that the requested module has been loaded into the operating system memory and installed;
if the module has not been loaded into the operating system memory and installed into the operating system, loading the module comprising the steps of;
copying compiled module code of the requested module into the operating system memory such that the requested module is a loaded module, and
resolving all references between the loaded module and the at least one loaded and installed module,
installing the loaded module by providing references to the loaded module necessary for requests for access thereby indicating that the requested module has been loaded and installed, wherein the loaded module is now accessible, and
permitting the intercepted request for access to process; and
if the module has been loaded into the operating system memory and installed into the operating system, immediately executing the request for access to execute. (Main Claim)

1. A computer system comprising a central processing unit (CPU) and memory, said CPU executing in accordance with an operating system comprising modules of code, said computer system comprising a module sub-system for dynamically loading modules of the operating system

into operating system memory and installing loaded modules as part of the operating system for access, said subsystem comprising:
a control module loaded into operating system memory and installed as part of the operating system, said control module intercepting a request for access to a module such that the request is not performed, said control module further determining whether the requested module has been loaded and installed;
a loader module loaded into operating system memory and installed as part of the operating system, said loader module coupled to the control module for copying compiled module code for the requested module into the operating system memory to provide a loaded module;
a linker module loaded into operating system memory and installed as part of the operating system, said linker module coupled to the control module for resolving the references between the loaded module and at least one module previously loaded in the operating system memory and installed;
an install module loaded into operating system memory and installed as part of the operating system, said install module coupled to the control module for providing references to the loaded module thereby indicating that the loaded module is a loaded and installed module;
if said control module determines that the requested module has not been loaded and installed, said control module controlling the loader module, linker module and install module respectively to load, link and install the requested module, said control module further permitting the request for access to execute once the requested module has been loaded and installed; and
if said control module determines that the requested module has been loaded and installed, said control module immediately permitting the request for access to execute.

2. The computer system as set forth in claim 1, wherein the operating system further comprises at least one module configuration table comprising information regarding modules, said control module reading the module configuration table to determine whether the requested module has been loaded into the operating system memory and installed.
3. The computer system as set forth in claim 2, wherein each requested module comprises a wrapper to provide information indicating whether the module is a loadable file, a type of module, and a pointer to installation code located in the operating system for the type of module, said installation code providing specific instructions as to the installation of the module information in the configuration table.
4. The computer system as set forth in claim 1, wherein said module configuration table comprises a record for each module, each record comprising at least one entry having a value indicating that the module is loaded in the operating system memory and installed.
5. The computer system as set forth in claim 1, wherein said loader module determines the size of a file of compiled module code, allocates operating system memory for placement of the file and writes the file in the memory allocated.
6. The computer system as set forth in claim 5, wherein if the loader module determines that insufficient operating system memory exists to write the file, a module is unloaded from the operating system to provide sufficient operating system memory.
7. The computer system as set forth in claim 4, wherein the control module determines if the requested module is loaded and installed by determining if the predetermined entry in the record contains a value indicative that the module is loaded into the operating system and installed.
8. The computer system as set forth in claim 4, wherein said install module allocates a record in said module configuration table for the requested module when the requested module is first loaded into the operation system memory and installed.
9. The computer system as set forth in claim 4, wherein a record in the module configuration table is preallocated for each module that can be loaded and installed.
10. The computer system as set forth in claim 4, wherein the install module establishes a record in the module configuration table for the

loaded module when the loaded module is first installed in the operating system, said record comprising the entry containing the value indicative that the loaded module is installed in the operating system.

11. The computer system as set forth in claim 4, wherein a plurality of modules are loaded and installed and said module subsystem further comprises:
means for selecting a loaded and installed module to unload and uninstall from the operating system;
means for nullifying the entry, that indicates that the selected module is loaded in the operating system and installed, in the module configuration table such that the entry in the module configuration table indicates that the selected module is no longer loaded or installed;
means for reallocating the operating system memory occupied by the selected loaded and installed module to unload such that the memory can be utilized to load another module into the operating system.

12. The computer system as set forth in claim 10, wherein said control means determines if the requested module is loaded and installed by determining if a record exists for the requested module in the module configuration table and if a record exists, if the entry in the record contains the value indicative that the module is loaded and installed in the system.

14. The process for dynamically configuring an operating system as set forth in claim 13 wherein a record is allocated for a module when a modules is first loaded and installed, said step of determining further comprising the step of ascertaining whether a record for the requested module exists, wherein if a record for the requested module does not exist or if a record for the requested module exists and the entry indicates the requested module is not loaded and installed, the module is loaded and installed.

15. The process for dynamically configuring an operating system as set forth in claim 13, further comprising the steps of:
selecting a loaded and installed module to unload and uninstall from the operating system;
nullifying the entry for the selected module in the module configuration table to indicate that the selected module is uninstalled; and
reallocating the operating system memory occupied by the selected module to unload such that the memory can be utilized to load another module into the operating system.

16. The process for dynamically configuring an operating system as set forth in claim 13, further comprising the steps of:
selecting a module to unload from the operating system, said module being in an uninstalled state, as indicated by a null entry in the entry for the selected module in the module configuration table; and
reallocating the operating system memory occupied by the selected module to unload such that the memory can be utilized to load another module into the operating system.

17. The process for dynamically configuring an operating system as set for in the claim 16 wherein said process further comprises the step of uninstalling at least one installed and loaded module according to predetermined parameters by nullifying the entry for the installed and loaded module in the module configuration table.

18. In a computer system comprising resources such as a Central Processing Unit (CPU), memory and input/output means, an operating system comprising a kernel memory comprising a plurality of installed and loaded modules which execute processes and utilize the resources, a process for loading modules into the kernel memory on an as needed basis comprising computer implemented steps of:
compiling each module into an object code format;
storing the compiled files representative of the modules in a memory external to the kernel memory;
providing at least one module configuration table in the kernel memory identifying modules loaded into the kernel memory and installed into the kernel as part of the operating system;
when a requesting process issues an access request for a requested module, said process for loading,

intercepting the access request to the module configuration table;
reviewing the entries in the module configuration table to determine if the requested module has been loaded and installed into the kernel,
if the module configuration table indicates that the module has not been loaded and installed into the kernel,
reading the compiled file representative of the module,
determining the size of the file,
allocating space in the kernel memory for the file,
copying the file into kernel memory,
resolving all references between the requested module and the loaded and installed modules,
installing the module by identifying the location of the loaded module and updating the module configuration table to indicate that the module has been loaded and installed into the kernel, and
permitting the request to access to execute; and
if the module configuration table indicates that the module has been loaded and installed into the kernel, immediately permitting the request to access to execute.

5/5/34 (Item 3 from file: 654)

3849167 **IMAGE Available
Derwent Accession: 1997-297649
Utility
REASSIGNED
E/ **Software mechanism for accurately handling exceptions generated by speculatively scheduled instructions**

Inventor: Adler, Michael C., Lexington, MA
Hobbs, Steven O., Westford, MA
Lowney, Paul G., Concord, MA
Assignee: Digital Equipment Corporation (02), Maynard, MA
Digital Equipment Corp (Code: 23989)
Examiner: Harvey, Jack B. (Art Unit: 235)
Assistant Examiner: Pancholi, Jigar
Combined Principal Attorneys: Drozenski, Diane C.; Maloney, Denis G.; Fisher, Arthur W.

	Publication Number	Kind	Date	Application Number	Filing Date
Main Patent	US 5634023	A	19970527	US 94270184	19940701
Priority				US 94270184	19940701

Current US Classification (Main): 712244000 (X-ref): 712023000
US Classification on document (Main): 395591 (X-ref): 395800
International Classification (Edition 1): G06F-009/30; G06F-015/16
Examiner Field of Search (US): 395800; 395375; 395700; 395390; 395391; 395591

Cited US Patents:

Patent Number	Date YYYYMM	Main US Class	Inventor
US 5421022	199505	395800	McKeen

Fulltext Word Count: 7400
Number of Claims: 21
Exemplary Claim Number: 1
Number of Drawing Sheets: 10
Number of Figures: 11
Number of US cited patent references: 1

Post Issue Legal Status:
Calculated Expiration Date: 20140701

Reassignment:
Recorded: 20020109
Action: ASSIGNMENT OF ASSIGNORS INTEREST
Assignor: DIGITAL EQUIPMENT CORPORATION DATE SIGNED: 12/09/1999
COMPAQ COMPUTER CORPORATION DATE SIGNED: 06/20/2001
Assignee: COMPAQ INFORMATION TECHNOLOGIES GROUP, L.P. 20555 STATE HIGHWAY 249 HOUSTON TEXAS 77070
Reel: 012447
Frame: 0903
Contact: CONLEY, ROSE & TAYON, P.C. JONATHAN M. HARRIS P.O. BOX 3267 HOUSTON, TEXAS 77253-3267

Abstract:
Methods for handling exceptions caused by speculatively scheduled instructions or predicated instructions executed within a computer program are described. The method for speculatively scheduled instructions includes checking at a commit point of a speculatively scheduled instruction, a semaphore associated with the speculatively scheduled instruction and branching to an error handling routine in the semaphore is set. A set semaphore indicates that an exception occurred when the speculatively scheduled instruction was executed. For a predicated instruction the method includes checking a predicate of a eliminated branch and a semaphore associated with the speculative instruction at a commit point of the speculative instruction and branching to an error handing routine if the semaphore indicates that an exception occurred when the speculative instruction was executed, and the predicate is true, which indicates that the speculative instruction was properly executed.

What is claimed is:
1. A method for handling an exception caused by a speculatively scheduled instruction which is executed within a computer program, said method comprising the steps of:
checking at a commit point of said speculatively scheduled instruction, a semaphore associated with said speculatively scheduled instruction; and
branching to an error handling routine if the semaphore indicates that an exception occurred when said speculatively scheduled instruction was executed. (Main Claim)
2. The method of claim 1 including the step of:
continuing execution of the computer program if said semaphore indicates that an exception did not occur.
3. The method of claim 1 wherein prior to said checking step, the method includes the step of:
executing said speculatively scheduled instruction in a program; and
detecting whether an exception occurs.
4. The method of claim 3 further including the step of:
noting the occurrence of the exception caused by the execution of said speculatively scheduled instruction.
5. The method of claim 4 wherein the step of noting the occurrence of the exception includes the steps of:
determining a program counter value of said speculatively scheduled instruction;
mapping said program counter value to a semaphore; and
setting said semaphore associated with said speculatively scheduled instruction.
6. The method of claim 5 wherein said step of mapping further includes:
locating a descriptor corresponding to said program counter value wherein said descriptor identifies said semaphore.
7. The method of claim 5 wherein said step of setting further includes:
modifying a bit in a general register.
8. The method of claim 1 further including the step of:
executing an additional non-speculative instruction; and
detecting whether an exception occurs in response to executing said additional non-speculative instruction.
9. The method of claim 8 further including the step of:
noting the occurrence of the exception caused by the execution of

said additional non-speculative instruction.

10. The method of claim 9 wherein the step of noting the occurrence of the exception includes the steps of:
determining a program counter value of said additional non-speculative instruction;
mapping said program counter value to a semaphore; and
raising the exception immediately if said program counter is not associated with a semaphore.
11. The method of claim 10 wherein said step of mapping further includes:
searching a program counter array table to locate descriptor corresponding to said program counter value wherein said program counter array table or said descriptor does not associate a semaphore with said program counter value.
12. A computer language translator which speculatively schedules instructions comprising:
means for translating a program into a target language;
means for reordering an instruction within said target language program from the order specified in the program said reordered instruction being referred to as a speculatively scheduled instruction; and
means for associating a semaphore with said speculatively scheduled instruction.
13. The translator of claim 12 further including:
means for including in an executable image of said program, instructions to test said semaphore after a commit point of said speculatively scheduled instruction.
14. The translator of claim 13 wherein said translator further includes:
means for including in an executable image of said program instructions to report an exception if said test of said semaphore indicates an exception occurred when said speculatively scheduled instruction was executed.
15. The translator of claim 12 further including:
means for including in an executable image of said program instructions to initialize said semaphore prior to executing said speculatively scheduled instruction.
16. The translator of claim 12 wherein said semaphore is represented as a pattern of at least one bit in a general register.
17. The translator of claim 12 wherein said means for associating further comprises:
means for including in an executable image of said program a data structure which maps said speculatively scheduled instruction to said semaphore.
18. The translator of claim 17 wherein said data structure maps a program counter of said speculatively scheduled instruction to said semaphore.
19. The translator of claim 12 wherein said means for associating includes:
a program counter array table;
a descriptor; and
wherein said program counter table has a pointer field which is used to locate said descriptor and said descriptor has a number representing a bit within a general register of a processor on which translated code is run.
20. The translator of claim 19 wherein said program counter array table includes two fields:
a program counter field; and
said pointer field which points to said descriptor.
21. The translator of claim 20 wherein said descriptor includes:
a number representing a displacement of said speculative instruction from said program counter; and
said number representing a bit within said general register wherein said bit is said semaphore associated with said speculative instruction.

3842709 **IMAGE Available
Derwent Accession: 1997-271643
Utility
REASSIGNED
E/ Software mechanism for accurately handling exceptions generated by instructions scheduled speculatively due to branch elimination
Inventor: Adler, Michael C., Lexington, MA
Hobbs, Steven O., Westford, MA
Lowney, Paul G., Concord, MA
Assignee: Digital Equipment Corporation (02), Maynard, MA
Digital Equipment Corp (Code: 23989)
Examiner: Lim, Krisna (Art Unit: 235)
Combined Principal Attorneys: Drozenski, Diane C.; Maloney, Denis G.; Fisher, Arthur W.

	Publication Number	Kind	Date	Application Number	Filing Date
Main Patent	US 5627981	A	19970506	US 94270192	19940701
Priority				US 94270192	19940701

Current US Classification (Main): 712235000 (X-ref): 712244000
US Classification on document (Main): 395582 (X-ref): 395591
International Classification (Edition 1): G06F-009/38
Examiner Field of Search (US): 395375

Cited US Patents:

Patent Number	Date YYYYMM	Main US Class	Inventor
US 5420990	199505	395375	McKeen
US 5421022	199505	395800	McKeen
US 5452426	199509	395375	Papworth
US 5463745	199510	395375	Vidwans
US 5465336	199511	395375	Imai

Fulltext Word Count: 7546
Number of Claims: 12
Exemplary Claim Number: 1
Number of Drawing Sheets: 10
Number of Figures: 11
Number of US cited patent references: 5
Post Issue Legal Status:
Calculated Expiration Date: 20140701

Reassignment:
Recorded: 20020109
Action: ASSIGNMENT OF ASSIGNORS INTEREST
Assignor: DIGITAL EQUIPMENT CORPORATION DATE SIGNED: 12/09/1999
COMPAQ COMPUTER CORPORATION DATE SIGNED: 06/20/2001
Assignee: COMPAQ INFORMATION TECHNOLOGIES GROUP, L.P. 20555 STATE HIGHWAY 249 HOUSTON TEXAS 77070
Reel: 012447
Frame: 0903
Contact: CONLEY, ROSE & TAYON, P.C. JONATHAN M. HARRIS P.O. BOX 3267 HOUSTON, TEXAS 77253-3267

Abstract:
Methods for handling exceptions caused by speculatively scheduled instructions or predicated instructions executed within a computer program are described. The method for speculatively scheduled instructions includes checking at a commit point of a speculatively scheduled instruction, a semaphore associated with the speculatively scheduled instruction and branching to an error handling routine if the semaphore is set. A set semaphore indicates that an exception occurred when the speculatively scheduled instruction was executed. For a predicated instruction the method includes checking a predicate of an eliminated branch and a semaphore associated with the speculative

instruction at a commit point of the speculative instruction and branching to an error handling routine if the semaphore indicates that an exception occurred when said speculative instruction was executed, and the predicate is true, which indicates that said speculative instruction was properly executed.

What is claimed is:

1. A method for handling an exception caused by a speculative instruction provided for an eliminated conditional branch, said speculative instruction is executed within a computer program, said method comprising the steps of:
   checking a predicate of said eliminated branch and a semaphore associated with said speculative instruction at a commit point of said speculative instruction;
   branching to an error handling routine if the predicate indicates that said speculative instruction should have executed, and the semaphore indicates that an exception occurred when said speculative instruction was executed. (Main Claim)
2. The method of claim 1 wherein the step of checking further includes the substep of:
   selecting between a register filled with clear semaphore values and a semaphore register which includes said semaphore, based upon a value of a predicate from said eliminated branch.
3. The method of claim 2 including the step of:
   continuing execution of the computer program if said semaphore indicates that an exception did not occur.
4. The method of claim 2 wherein prior to said checking step, the method includes the step of:
   executing said speculative instruction in a program; and
   detecting whether an exception occurs.
5. The method of claim 4 further including the step of:
   noting the occurrence of the exception caused by the execution of said speculative instruction.
6. The method of claim 5 wherein the step of noting the occurrence of the exception includes the steps of:
   determining a program counter value of said speculative instruction;
   mapping said program counter value to a semaphore; and
   setting said semaphore associated with said speculative instruction.
7. The method of claim 6 wherein said step of mapping further includes:
   locating a descriptor corresponding to said program counter value wherein said descriptor identifies said semaphore.
8. The method of claim 6 wherein said step of setting further includes:
   modifying a bit in a general purpose register.
9. The method of claim 1 further including the step of:
   executing an additional non-speculative instruction; and
   detecting whether an exception occurs in response to executing said additional non-speculative instruction.
10. The method of claim 9 further including the step of:
    noting the occurrence of the exception caused by the execution of said additional non-speculative instruction.
11. The method of claim 10 wherein the step of noting the occurrence of the exception includes the steps of:
    determining a program counter value of said additional instruction;
    mapping said program counter value to a semaphore; and
    raising the exception immediately if said program counter is not associated with a semaphore.
12. The method of claim 11 wherein said step of mapping further includes:
    searching a program counter array table to locate a descriptor corresponding to said program counter value wherein said program counter array table or said descriptor does not associate a semaphore with said program counter value.

4/9/11 (Item 2 from file: 8)
DIALOG(R)File 8:Ei Compendex(R)

01094595 E.I. Monthly No: EI8202010282 E.I. Yearly No: EI82020146
Title: DESIGN OF A UNIVERSAL MICROPROCESSOR LINKER.
Author: Wirfs-Brook, Rebecca J.
Corporate Source: Tektronix Inc, Beaverton, Oreg, USA
Source: Mini and Microcomput, v 5 n 3 Publ by ACTA Press, Anaheim, Calif, USA, and Zurich, Switz, 1980 p 100-105
Publication Year: 1980
Language: ENGLISH
Journal Announcement: 8202
Abstract: This study focuses on the issues and complexities of designing a universal, **relocating linking** system. The way that microprocessors are programmed and configured in a typical application influences the universal linking system design. Microprocessor features that affect memory relocation and allocation strategies must be identified, categorized, and incorporated in the linking system. Seemingly minor differences in microprocessor architectures can profoundly impact a universal linker. A model is presented of microprocessor memory usage that encompasses all of these considerations. The design of a **relocating linking** system that incorporates this model is presented. Particular emphasis is placed on the memory-classification scheme and memory-allocation algorithms that were developed. 3 refs.
Descriptors: *COMPUTERS, MICROPROCESSOR
Identifiers: MICROPROCESSOR LINKER
Classification Codes:
723 (Computer Software)
72 (COMPUTERS & DATA PROCESSING)